Natural Language Interactions with Artificial Experts

TIMOTHY W. FININ, MEMBER, IEEE, ARAVIND K. JOSHI, FELLOW, IEEE, AND BONNIE LYNN WEBBER, MEMBER, IEEE

Invited Paper

The aim of this paper is to justify why Natural Language (NL) interaction, of a very rich functionality, is critical to the effective use of Expert Systems and to describe what is needed and what has been done to support such interaction. Interactive functions discussed here include defining terms, paraphrasing, correcting misconceptions, avoiding misconceptions, and modifying questions.

I. INTRODUCTION

Natural Language (NL) interfaces to database systems are already proving their worth. They allow users to get at the database facts they want, without the need to become system wizards. In this paper we are primarily concerned with NL interfaces for systems that do more than identify and retrieve facts. Such systems, often called knowledge based systems, expert systems, or advisory systems, are expected to provide analyses and/or advice to users faced with real problems. Our main goals in this paper are to justify why NL interaction, of a very rich functionality, is critical to the effective use of these systems and to demonstrate what is needed to support it. Even if one wanted to employ a formal language or menu and pointer-based system for this role, it would have to have many of the features of NL that allow the kinds of interactive functions that a system must support.

Returning for a minute to database systems, to naive or infrequent users of these systems, NL can mean getting their information simply by asking for it. In many cases, NL can even mean a shorter, simpler query than its formal counterpart [51]. "Smarts" built into NL front ends can increase their tolerance for typing errors, spelling errors, and divergences from polished grammar, or eliminate such errors entirely [67]. Other "smarts" can increase the scope of acceptable NL input beyond "syntactically sugared" formal queries by allowing the user to talk about things previously mentioned in either question or answer without previously having named them (via the use of anaphoric and deictic pronouns and noun phrases such as "them," "this," "those students," "the seniors," etc.). On the basis of such improvements in performance characteristics, NL interfaces for databases are becoming more attractive, even where the interactive behaviors required are not very sophisticated.

However, our claim is that for effective use of Expert Systems, NL is not just attractive but critical. This is because advice and diagnosis cannot be trusted blindly. From what we know of human–human advisory (problem-solving) interactions, a responsible person will not accept advice if s/he cannot request and receive clarification, e.g.,

> System: In your agreement of sale, make sure that you have an engineering inspection clause.
> User: Engineering inspection clause?
> System: Right. You want that house inspected by an engineer just to make sure that there's nothing wrong with it that you might have missed. The clause stipulates that an engineer be permitted at your expense to inspect the house and that if there is anything in that house that requires a repair (and you can set the amount, $250 or $500), the owner will be responsible for it, or else you will be permitted to get out of the deal.

if s/he cannot verify that s/he has understood the advice or diagnosis correctly, e.g.,

> Expert: OK, under those circumstances, I would not object to see you go back into that for another six months.
> User: So you roll it over, in other words?
> Expert: Right.

if s/he cannot get justification as to why that advice or diagnosis should be accepted, e.g.,

> Expert: So you would still file as a single person.
> User: Even though I'm married?

Manuscript received January 27, 1986; revised March 7, 1986. This work was supported by the U.S. Army under Grants DAAG-29-84-K-0061, DAAB-07-84-K-FO77, DAAG-29-84-9-0027, DAAG-29-85-K-0061; by DARPA under Grant N00014-85-K-0018, and by the National Science Foundation under Grants MCS-82-19116-CER, MCS-82-07294, MCS-83-05221, and DCR-84-10413.

The authors are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA.

0018-9219/86/0700-0921$01.00 ©1986 IEEE

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUL 1986		**00-07-1986 to 00-07-1986**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Natural Language Interactions with Artificial Experts	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Pennsylvania,Department of Computer and Information Science,Philadelphia,PA,19104**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**18**	

Expert: Right. Because you weren't married during the filing year, 1984.

or if s/he cannot add or delete parts of the problem specification (i.e., symptoms of the problem or constraints on its solution), e.g.,

User: What if I decided to postpone my wedding until January 1?

The complexity of this whole enterprise requires a rich communicative system such as NL (and currently nothing else) provides.

Moreover, there is another problem with achieving successful communication that makes NL not just attractive but critical. Suchman [63] describes the situation as follows:

> Successful communication under ordinary circumstances is not a product of the absence of trouble but of its repair. That is, the extent to which we are able to communicate effectively is not a function of our planning for all of the contingencies that might arise. It is the function of the locally managed, collaborative organization of the interaction itself—an organization designed to maximize efficiency by providing for the ongoing identification and repair of troubles that, when we are constructing our actions together, in real time, inevitably arise.

The kinds of interactions we will be concerned with are well-characterized by this observation. One reason for this is that we do not believe that we can count on users having well-formed problems, or an accurate understanding of the system's capabilities and terminology, or even a good understanding of the domain. Under such conditions, a formulaic, trouble-free interaction through a menu/pointer system or a formal language is inconceivable, assuming as it does that users can state their problems right off, receive a response, and leave, confident that applying this response to their problem will result in success. In what follows, we discuss specific NL capabilities that can help user and system to come to terms with each other. If such capabilities are not supported, any "user friendliness" intended in the system design (through NL input and output, for example) will be irrelevant. The appealing "syntactic sugar" will turn out to be "syntactic arsenic" in disguise.[1] Our position is that support for these NL capabilities cannot come solely from the NL interface itself: that in many cases, the underlying reasoning system and knowledge base will have to be adapted or designed *ab ovo* to support them. On any grounds, the latter is clearly preferable.

In this paper, we will not discuss basic issues involved in developing NL interfaces for database fact-retrieval systems (e.g., the syntactic and semantic analysis of user requests, resolving anaphora and ellipses, mapping between user concepts and an underlying database query language, etc.). One reason is that by now there exist some fine surveys of such work, which are quite up-to-date and easily accessible. For example, Perrault and Grosz [52] survey NL interface systems from the point of view of system architecture and development tools. Bates and Bobrow [4] describe the state of the art of NL interfaces for databases, circa 1984.

The other reason for ignoring these more basic issues is that we want to focus on other aspects of cooperative interactions where NL interfaces will really pay off. Some of the extensions to NL interfaces that we will be discussing in the following sections have, in fact, been developed in the context of database systems. However, the reason we have included them is the payoff they will have for interactions with knowledge based systems, expert systems, etc. (It is worth noting that as database systems are augmented with more and more general knowledge about their domains, as well as specific facts, the line between database system and knowledge based system becomes faint indeed.) For the reader's benefit though, we list here some of the major NL interface systems for databases: LUNAR [72], REQUEST and TQA [15], [53], REL [67], PLANES [70], NL Menu System [67], EXPLORER [37], RUS [5], [4], and a (nameless) system built by Ginsparg [22]. Some of the commercial systems are INTELLECT (Artificial Intelligence Corporation), THEMES (Frey Associates), RAMIS (Mathematica Products Group), CLOUT (Microrim), PLUME (Carnegie Associates), EXPLORER (Cognitive Systems), among others as reported in TARGET (*Artificial Intelligence Newsletter*, vol. 1, no. 3, May 1985).

There is one significant type of cooperative behavior needed for interacting with expert systems and knowledge based systems that we do not discuss here: that is, explanation, in the sense of justification: e.g.,

Tax Advisor: So you would still file as a single person.
Taxpayer: Even though I'm married?
Tax Advisor: Right, because you weren't married during the filing year, 1984.

Systems must be able to justify their actions—whether they act to ask a question or to offer some conclusion or advice (as above)—or users may not be sufficiently convinced to respond appropriately (i.e., to answer the system's question or to take its advice).[2] Here we just point the reader to some work on enabling systems to produce explanations which are accurate (i.e., they correctly specify the reason for the system's behavior) [64], [50], [16], clear [26], [58], and appropriate to their users' interests and/or level of expertise [44] [69].

The next section (Section II) describes two preliminary efforts aimed at giving systems the ability to automatically construct useful definitions of the terms they use. It also makes recommendations for further work in this area. Section III describes work that has been done on giving systems the ability to paraphrase their users' queries, allowing their users to verify how their queries have been understood. It also makes recommendations for further work. Section IV deals with recognizing and responding to disparities between the user's view of the world and that of the system. We describe some work that has been done in the context of database interfaces and some more applicable to interfacing with expert and knowledge based systems. Section V discusses research on the complementary issue of avoiding misleading remarks which may lead to new misconceptions on the user's part. And finally, Section VI discusses how troubles can be avoided or lessened in

[1]Note that this research also helps in isolating aspects of cooperative interaction which may be incorporated in a more formal system, to provide some of the flexibility available in NL, if that is the path one wishes to follow.

[2]Notice that this sense of "explanation" is different from its sense of "clarification," as in "Explain what you mean by X." We discuss clarifying terminology in Section II.

determining the user's problem, by having systems adapt their questions and interpretations to the user. We conclude in Section VII with a summary prognosis and recommendations.

II. Clarifying Terminology

To avoid troubles or as one way to repair them, user and system must establish a single terminology for interacting with each other. Without a single mutually understood terminology, the user will not be able to understand what the system is requesting or recommending; e.g.,

System: Is renal function stable?
User: What do you mean by 'stable'?

System: In your agreement of sale, make sure that you have an engineering inspection clause.
User: Engineering inspection clause?
System: Right. You want that house inspected by an engineer just to make sure that there's nothing wrong with it that you might have missed. The clause stipulates that an engineer be permitted at your expense to inspect the house and that if there is anything in that house that requires a repair (and you can set the amount, $250 or $500), the owner will be responsible for it, or else you will be permitted to get out of the deal.

nor will the expert system be able to understand what the user is requesting or reporting, comparable to the following exchange:

Medical Student: These are just stasis changes.
Expert Diagnostician: What do you mean by that?
Medical Student: Just from chronic venous insufficiency.

If the user and expert system are to establish a single terminology for the interaction, there are three options—either the system understands and accepts the user's terminology, the user understands and accepts the system's, or together they accept a third. Here we only want to consider the first two options.

Now it has been claimed [10] that in the case of a physician (i.e., a "natural" expert) interacting with a patient to take a history, it is an error for the physician to use his or her term for a symptom rather than the patient's—that patients seem to have difficulty characterizing their experiences in the physician's terms.[3] The physician must accommodate to the patient's terminology and, where necessary, seek clarification from the patient. For example, in the following exerpts from a history-taking session, the physician first attempts to verify that he and the patient mean the same thing by 'hemorrhoids' and later asks the patient directly what he means by 'diarrhea' [10].

Physician: How can I help you?
Patient: OK. I've had bleeding right before my bowel movement. Some diarrhea and then bleeding.
Phys: Mm-hm. And when is the first time that ever happened?
Pat: I really don't remember because I didn't make any note of it. It was just like a little diarrhea and no blood—I think I had hemorrhoids.
Phys: I see. And how did you know it that time that you had hemorrhoids?
Pat: I thought they were because I felt something there.
Phys: Did you feel something? A lump or something like that?
Pat: Yeah, like two little lumps.
Phys: Right. Now. Did the diarrhea come first?
Pat: Yeah.
Phys: And when was that? Roughly speaking.
Pat: About a month ago, maybe—but then it was on and off I—and I did—
Phys: You mean for one day you would have dia—a loose—*What do you mean by diarrhea?*
Pat: Not diarrhea. I don't even know. Like, I can't describe it—Like I would feel very gassy. And, like, I guess expel—I don't know if it was mucus or whatever.

This would seem to argue for the expert system adapting to the user's terminology—at least in attempting to acquire information from the user. Moreover, in later explaining any of its conclusions, it has been argued that a system must make its explanation understandable to the user—one element of which is that the explanation not contain terms that the user will not understand.

On the other hand, others have noted [12], [48], [39] that people are notoriously poor at defining terms in a way that a system can understand and use. This is well illustrated in the case of NL interfaces confronted with new terms that appear in database queries, that are not in their lexicons or grammars. There are currently four viable methods for dealing with such 'unknown' terms: the first, used in [27], is to let the user equate the new term or phrase with one the system already knows (which assumes solid knowledge of the system on the user's part); e.g.,

User: Who commands the *Kennedy*?
System: I don't know the word "command."
User: Let 'who commands the *Kennedy*' be the same as 'who is the captain of the *Kennedy*'
System: OK
User: Who commands the *Kennedy*?

The second method is to guide the user step by step through a simple defining process [48], [39] to acquire both syntactic and semantic information about the new term. If the word cannot be defined by this process, the user's only recourse is asking the system programmer to do it or forgetting about the term. It would be foolish to ask users

System: What do you mean by 'concerning'?
System: What do you mean by 'often'?
System: What do you mean by 'purple'?

[3]On the other hand, if the patient seems to lack a way to describe his/her experience (say, of pain), Cassell would have the physician suggest terms which may evoke a useful response from the patient.

and expect them to come up with a response that makes sense and is usable by the system.

The third method is to assume that the unknown term is a database value (a reasonable practice in large, rapidly changing systems where one does not want to put all database values with their syntactic and semantic features into the lexicon) and either wait until after query evaluation or verify understanding through query paraphrase (see next section) to find out whether this is a reasonable assumption. This is the method used in INTELLECT [25].

The fourth option is to punt entirely and ask the user to rephrase the query without the offending terms.

The upshot of all this is that it is not clear to us who should adapt to whom. If the expert system is to adapt to users' terminology (both in acquiring information from them and in explaining its conclusions to them), it will need to develop better ways of figuring out what the user means by particular terms. If a user is to adapt to the system's terminology, the system becomes responsible for explaining its terms to the user in ways that s/he will understand and find relevant. In most cases, this will be impossible to do without the use of freely generated language. So far, two attempts have been made at enabling systems to take this responsibility through the incorporation of domain-independent NL definitional capabilities. Both of these are described below.

A. TEXT

TEXT [45] was a first attempt at providing a domain-independent NL definitional capability which can explain to a user the system's understanding of the terms it uses. It does this by piggybacking its definitional machinery on an augmented version of the system's domain model. If the user requests the definition of a term or an explanation of the difference between two terms, TEXT

- circumscribes an area of relevant information that it might include in its response. (Not all this information need be included in the response, but no information other than this will be included.);
- chooses an appropriate, fairly flexible format for presenting the information (called a "schema");
- instantiates the schema with a subset of the relevant information, guided both by the options allowed in the schema and a desire to allow the 'focus' of the text to change in a natural way;
- produces NL text from the instantiated schema.

The kind of information represented in TEXT's augmented data model in order to produce useful definitions of and comparisons between terms includes

- simple taxonomic information (e.g., a SHIP is a WATER_VEHICLE);
- for each split in the taxonomy, the criterion for the split (called the "distinguishing descriptive attribute" or DDA). Each subtype in a partition has the same DDA name, but a different value for the attribute (e.g., WATER_VEHICLE has two subtypes SHIP and SUBMARINE. SHIP has its DDA, TRAVEL_MODE, equal to SURFACE, while SUBMARINE has TRAVEL_MODE equal to UNDERWATER);
- for each subtype in a partition, the database attributes that support its DDA (e.g., SHIP has DRAFT and DISPLACEMENT). These are called "supporting database attributes";
- for each subtype, a list of its other database properties (e.g., SHIP has additional properties PROPULSION, MAST_HEIGHT, MAXIMUM_SPEED, etc.).

TEXT's presentational schema were derived from empirical observation of short (one paragraph) definitions and term comparisons. (Current schema include IDENTIFICATION, CONSTITUENCY, COMPARE_&_CONTRAST, ATTRIBUTIVE.) Each schema corresponds to a regular grammar whose terminals are *rhetorical predicates* which themselves specify the kind of information from the system's data model that can instantiate them. For example, the IDENTIFICATION schema has the following structure:

Identification (Class & Attribute/Function)
{Analogy/Constituency/Attributive/Amplification}*
Particular_Illustration/Evidence +
{Amplification/Analogy/Attributive}
{Particular_Illustration/Evidence}

where "{ }" indicates optional constituents, "/" indicates alternatives, "+" indicates that the item may appear one or more times, while "*" indicates that the item is optional, but if present, may appear one or more times. With respect to the information in the system's data model, instantiating the rhetorical predicate "Attributive" with respect to a concept can be done via its DDA or its other database attributes. "Particular_Illustration" can be instantiated by any of the terminal elements below a concept in the data model. The following definition of AIRCRAFT_CARRIER produced by TEXT instantiates this IDENTIFICATION schema as the sequence of rhetorical predicates: Identification, Analogy, Particular_Illustration, Amplification, Evidence.

User: What is an AIRCRAFT_CARRIER?
TEXT: An aircraft carrier is a surface ship with a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063. Aircraft carriers have a greater LENGTH than all other ships and a greater DISPLACEMENT than most other ships. Mine warfare ships, for example, have a DISPLACEMENT of 320 and a LENGTH of 144. All aircraft carriers in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL, BEAM of 252,... A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULL_NO are CV.

The original version of TEXT [45] used a unification grammar [34] to translate an instatiated schema into NL text. The current version uses McDonald's MUMBLE [42] as its tactical component [57] and runs 60 times faster. TEXT is now being provided with a richer representation language for its domain model, which should allow it to make better judgements in its selection of relevant material and in how it instantiates its response.

B. CLEAR

While TEXT attempts to provide a portable, domain-independent NL definitional capability for users of database systems, CLEAR [57] attempts to do the same for users of

expert systems. It uses the information intrinsic to an expert system—i.e., its rule base—to explain to users how terms are used in the system. (There is really no intrinsic information that corresponds to what a term *means*.)

These term definitions should be of interest to both the system's intended users and its developers. That is, the larger the rule base, the greater the chance that the same term may be used in more than one way (especially with rules being developed by more than one person or rules changing over a period of time). This is something that system developers must avoid. CLEAR's NL definitions can help them avoid this.

For a user, it is important to understand the system's terminology when s/he must answer the system's questions. Asking "why" the system asked a question (an option already allowed in many expert systems, following the work of Davis [16]) is unlikely to help him/her understand its terms: e.g., "filing status," "single," "married filing jointly," etc, in the following question:

System: What is your filing status?
1—Single
2—Married filing separately
3—Married filing jointly
4—Unmarried head of household
User: WHY?
[i.e., WHY is it important to determine your filing status?]
System: ...in order to compute total number of exemptions.

IF the filing status of the client is single
AND the number of special exemptions claimed by the client is *N*
THEN the total number of exemptions on the tax form is *N* + 1.

Of course, one could manually store definitions of all the terms used in the system, but again, this would require an enormous amount of manual labor, with the additional difficulty of keeping the definitions up to date as the system develops.

In deciding whether to use a rule in explaining a given concept, CLEAR ranks it according to how it uses the concept. For example, a rule in which a concept appears in the premise as a property or its value (provided it does not also appear in the conclusion) is ranked HIGH with respect to that concept, while a rule in which that concept appears as a property whose value is inferred is ranked MEDIUM, and one in which the concept appears as a concluded value is ranked LOW. This means that a rule such as R1 below will be rated higher than R2, with respect to explaining the term "filing status," since in R1, it appears as a premise property and in R2, as an inferred property.

R1: IF the filing status of the client is single,
AND the number of special exemptions claimed by the client is *N*,
THEN the total number of exemptions on the tax form is *N* + 1.

R2: IF the marital status of the client is unmarried or separated,
THEN the filing status on the tax form is single.

The actual translation of rules from their internal format into English is done through the use of pre-stored templates in the manner of MYCIN's translation methods [60]. However, if there are many rules to present, CLEAR provides two ways of condensing their presentation. First, if the rules have been organized into a generalization hierarchy, then each subset of the rules to be presented which falls under a single node of the hierarchy can be presented in terms of the single general rule; e.g.,

IF the filing status of the client is *X*
THEN the total number of exemptions of the client is determined from *X*.

The second method involves automatic summarization of subsets of the rules to be presented which share common concepts in both premise and conclusion—for example, if in explaining "filing status," several rules have "filing status" as a property in the premise and "total number of exemptions" as an inferred property in the conclusion, this would be summarized as

Filing status can be used to determine the total number of exemptions.

(At the time of this writing, this summarization facility had not yet been implemented.)

A system such as CLEAR can only be as effective as its ability to recognize how terms are used in rules, without requiring additional knowledge of the domain.[4] In a MYCIN-like system, this can be done to some extent, but for really effective responses, more explicit rule structures are needed. For example, if the set of premise clauses in a rule *define* what it means for the concluded property to have a particular value, then that rule should actually be rated HIGH with respect to explaining that concluded value, as in

User: EXPLAIN head_of_household
System:
IF the client is unmarried on the last day of the year,
AND the client paid < 50% of the cost of keeping up the home of a parent or child who is a dependent of the client,
AND the client selects filing status head of household,
THEN the filing status of the client is head of household.

On the other hand, if the set of premise clauses set out symptomatic ways of *recognizing* whether the concluded property has a particular value, then that rule should be ranked LOW with respect to explaining the concluded value, as CLEAR currently does; e.g.,

User: EXPLAIN enterobacteriaceae
System:
IF the identity of the organism is not known with certainty,
AND the stain of the organism is gramneg,
AND the morphology of the organism is rod,
AND the aerobicity of the organism is aerobic,
THEN there is strongly suggestive evidence that the class of the organism is enterobacteriaceae.

[4]CLEAR is designed to be used with any expert system built using the HP-RL system [36] without altering the structure or content of the expert system's knowledge base.

Because CLEAR cannot distinguish between these two very different types of rules, it will make mistakes. But notice that the consequence of such mistakes is just a suboptimal definition, which includes material that the user may find irrelevant and omits some material that s/he may find useful. Thus a desire to be able to better clarify terms for users of expert systems argues for both an NL generation capability and more explicit structure to the system's knowledge base. Additional arguments for more explicit rule structures appear in [11] and have led to the development of MYCIN's successor NEOMYCIN.

C. Further Work

If systems are to be truly adept at clarifying the meaning of terms for their users, they will have to have the ability to

- adapt definitions to the current discourse context
- adapt definitions to users' previous knowledge and/or to their goals
- make use of pictures as well as text

With respect to adapting definitions to the current discourse context, McKeown [45] discusses the value that can be gained from discourse context defined as either 1) the user's previous requests for definitions or 2) those previous requests and their answers. (Neither sense of discourse context is stored or used in the original version of TEXT.) By just storing the user's previous requests for definitions, the following adaption can be made: if the user has previously asked about term A and now asks the difference between A and B, the system can omit information about A in its response (which it can presume to have given earlier) and just describe B with some concluding comparison between A and B. If the answer to the user's request for clarification of A had been stored as part of the discourse context as well, B can be presented in a way that mirrors the structure and content of the system's presentation of A. Also, if the user has asked about A and now asks about a similar concept B (where similarity can be defined with respect to the generalization hierarchy), parallels can be drawn between them and their similarities and differences discussed. If the answers to the user's requests have been stored as well, contrasts can be limited to those properties of A discussed previously. Otherwise any contrasts may be generated.

With respect to adapting definitions to users' previous knowledge and/or to their goals, an example of a goal-sensitive definition was given earlier:

System: In your agreement of sale, make sure that you have an engineering inspection clause.
User: Engineering inspection clause?
System: Right. You want that house inspected by an engineer just to make sure that there's nothing wrong with it that you might have missed. The clause stipulates that an engineer be permitted at your expense to inspect the house and that if there is anything in that house that requires a repair (and you can set the amount, $250 or $500), the owner will be responsible for it, or else you will be permitted to get out of the deal.

In this interaction, the user is asking the expert for advice on purchasing a house. In defining 'engineering inspection clause' here, the expert first shows how the concept is significant with respect to purchasing a house and then gives its formal definition. Depending on the current situation and the experience of the user, the significance of the concept to the current situation may be more or less evident from its formal definition alone, but to ensure that its significance is recognized, it should be included explicitly in the definition or the definition should be tailored explicitly to the goal. For example, compare the following three definitions:

User: DEFINE stainless_steel.

S1: Stainless steel is an alloy consisting of iron mixed with carbon, chromium, and often, nickel.

S2: Stainless steel is an alloy consisting of iron mixed with carbon, chromium, and often, nickel. It resists rust, stains, and corrosion caused by water and acid. On the other hand, a stainless steel blade does not take and keep a very sharp edge.

S3: Stainless steel is an alloy consisting of iron mixed with carbon, chromium, and often, nickel. It does not conduct heat efficiently, and the resulting hot and cold spots on the bottom of a stainless steel pan will not allow food to cook properly. Copper is a fast heat conductor and stainless steel pots are sometimes given copper bottoms to eliminate hot and cold spots.

Here, the user comes across the term "stainless steel," which s/he realizes s/he does not understand. If the user's goal in this situation is to buy kitchen knives, S1 and S3 would not be helpful definitions. If his/her goal is to buy pots, S1 and S2 would be equally as useless. In any goal-oriented interchange, a simple definition like S1 is rarely sufficient, and the significance of the term with respect to the goal must be made evident.

The importance of identifying and taking account of the speaker's plans and goals is becoming widely recognized both for understanding the speaker's utterances [1], [9], [38] and for generating cooperative responses [1], [31], [54], [61]. It should be applied to attempts to clarify meaning automatically as well.

Finally, systems should be able to make use of pictures (either pre-stored or automatically generated) in explaining terms or phrases to users. This is essential for 1) terms which name visual patterns, such as "spindle cell component" or "pseudopalisading" in neuropathology; 2) terms for objects and their parts which should be defined both visually (conveying structural information) and textually (conveying functional information); e.g., terms like "twing lines" and "boom vang," which refer to optional but useful parts of a racing dinghy;[5] or 3) terms or phrases which refer to actions; e.g., a term like "luff".[6]

[5] The 'boom vang' is a lever or block and tackle system used to hold the boom down and take the twist out of the mainsail. It should be anchored on the heel of the mast, as low as possible [14]. Notice that without a visual illustration, one would probably still not recognize a boom vang if it hit him/her on the head (and it usually does.)

[6] Luffing is when a leeward boat starts to turn into the wind, pushing up any windward boats beside it. One cannot luff beyond head to wind.

There has been some work in this area already. The APEX system [19] is an experimental system which creates pictorial explanations of action specifications automatically, as they are to be carried out in a particular context.[7] In addition, work in the Pathology Department at the University of Pennsylvania on a diagnostic expert system in neuropathology [62] projects the use of images (stored on an optical disk) to give users visual experience in concepts such as those mentioned above. If in working with the pathologist to diagnose a specimen, the system asks whether the specimen shows evidence of phenomenon *X*, the pathologist has the opportunity of seeing examples of *X* before answering. This is expected to enhance the reliability of the pathologist's response.

To summarize, we feel that it is critical in interactions with expert systems that user and system understand each other's terminology. Since this can rarely be established outside the context of the interaction—it is too much to ask of users that they learn the system's terminology beforehand—the interaction itself must provide for this functionality.

III. Paraphrase

The second place we see NL playing a critical role in users interacting with expert systems is in the use and understanding of paraphrase. Paraphrase—restating a text so as to convey its meaning in another way—is used in human–human interactions for three reasons, all attempting to ensure reliable communication.

- Where the listener notices an ambiguity or vagueness in the speaker's utterance, the listener can produce multiple paraphrases of the utterance, each corresponding to and highlighting a different alternative. (The set, of course, can be presented in an abbreviated way, to focus on the different options even more.) Given these presented alternatives, the speaker can easily indicate what was intended.
- Even without noticing an ambiguity or vagueness, the listener may attempt to paraphrase the speaker's utterance to show how it has been understood, looking for confirmation or correction from the speaker.
- The listener may paraphrase the speaker's utterance in order to confirm his or her belief that there is common understanding between them.

In each of these cases, a paraphrase must present a particular interpretation without itself introducing additional ambiguity.

The important thing about a paraphrase, whether to verify understanding or to point out the perceived ambiguity of the original utterance, is that it must be unambiguous, different in some obvious way from the original, and its meaning easily recognized. An early effort at paraphrasing wh-questions that conforms to these goals can be found in [43]. McKeown based her paraphrases on the linguistic notion of *given* versus *new* information. *Given* information is information either in the previous discourse or part of the shared world knowledge of the participants. In her database question/answering application (as part of the CO-OP system [33]), McKeown takes this to be information which the speaker assumes to be in the database. *New* information is of two types, that specifying what the speaker directly wants and that specifying the angle from which that want is to be satisfied. McKeown presents a simple algorithm for dividing the parse tree (syntactic analysis) of a question into three parts: 1) that conveying the new information directly wanted (the simple question, stripped of modifiers, containing the wh-phrase); 2) that conveying the interest angle of the new information (any syntactic modifiers of the wh-phrase); and 3) the assumed information (all the other modifiers). These parts are then rearranged and presented as an NL paraphrase to the user—assumptions, if any, first; direct question, second; and interest angle, if any, third. For example,

> Question: Which division of the computing facility works on projects in oceanography?
> Paraphrase: Assuming there are projects in oceanography, which division works on those projects? Look for a division of the computing facility.

If there is more than one modifier in a particular category, that information is presented in ways that avoid ambiguity. Additional assumptions are presented in parentheses after the first one; e.g.,

> Question: Which division works on projects in oceanography sponsored by NASA?
> Paraphrase: Assuming there are projects in oceanography (*those projects sponsored by NASA*), which division works on those projects?

Additional interest angles are presented in separate sentences at the end; e.g.,

> Question: Which division of the computing facility in superdivision 5000 works on projects in oceanography?
> Paraphrase: Assuming there are projects in oceanography, which division works on those projects? Look for a division of the computer facility. *That facility must be in superdivision 5000.*

Notice that if the system had parsed either question such that the additional modifiers were attached at other syntactically possible places, this would be evident from the paraphrase:

> Paraphrase: Assuming there are projects in oceanography (*that oceanography sponsored by NASA*), which division works on those projects?[8]
>
> Paraphrase: Assuming there are projects in oceanography, which division works on those projects? Look for a division of the computer facility. *That division must be in superdivision 5000.*

Thus although this system only produces paraphrases of one type of utterance—wh-questions, those paraphrases

[7] The STEAMER project makes excellent use of graphics to show cause–effect relationships in the control of a ship's steam plant, but not to explain terms [29].

[8] If this sounds like an impossible analysis, substitute the word "rockets" for oceanography.

are guaranteed both to differ from the original question, if the original contained any modifiers in the form of prepositional phrases or relative clauses (due to its separation of given and new information), and to be unambiguous (due to the way it presents multiple modifiers). McKeown's system produces paraphrases of the syntactic analysis of the user's question. However, McKeown notes that the paraphraser can also be made to generate an NL version of the system's interpretation and that this feature is used by CO-OP to produce a corrective response in case the user's question shows a misconception (cf. Section IV).

The point here is that there are places further down the line where the system's understanding of the user's query can become something other than the user intended. This can happen when relationships between items mentioned in the query are introduced explicitly. For example, in paraphrasing the following user query, Codd sees his proposed system RENDEZVOUS [12] as making explicit its interpretation of a dependency (grouping) relation among the items requested, as well as its interpretation of the relation implicit in the noun–noun compound "Houston parts":

User: Give me a list of all the part numbers, quantities, and suppliers of Houston parts.

... intervening clarifying interaction ...

System: This is what I understand your query to be: "Find all combinations of part number, quantity on order, supplier name, and supplier city such that the supplier supplied the part to a project located in Houston."

Notice that a dependency should not always be assumed—for example, in the similarly structured request like "Give me a list of the seniors and graduate students of the Computer Science Department." If the system has not previously established the validity of these interpretations through a clarification process, they may not turn out to be the ones intended by the user.

Also during subsequent interpretation, a term may be understood in ways the user does not expect or intend—for example, "rock" in the LUNAR system [71] was understood in a strict geological sense as meaning "basalt" rather than in its everyday sense.

Or terms may be particularized in ways the user does not intend or expect; e.g., the query

User: What do you have written by Jones?

may be paraphrased as

Paraphrase: What reports do you have written by Jones?

whereas the user meant any reports, books, articles, etc.

Or discourse context-sensitive items such as definite pronouns, definite noun phrases, and ellipses may be resolved in ways that the user has not intended.

User: Do you have the report that Finin published in 1984?

Paraphrase: *Test whether there are any reports whose author is Finin and whose publication date is 1984.*

System: No.

User: What reports do you have by Joshi?

Paraphrase: *List reports whose author is Joshi and whose publication date is 1984.*

Here the user's second request has been interpreted[9] as if the publication date had been ellipsed, and the system has supplied it explicitly as 1984 (i.e., the publication date from the previous request). This may or may not be what the user intended by his/her query, but the paraphrase makes the system's interpretion clear.

All of these argue for at least one additional point of contact where users can verify that the system has understood their queries as they intended them. At least two researchers have reported on systems which generate a paraphrase of the system's understanding of the user's utterance rather than a paraphrase of its syntactic analysis: an NL front-end to databases developed at Cambridge by Sparck-Jones and Boguraev [7], and an extension to IBM's TQA system developed by Mueckstein [49].

In Sparck-Jones and Boguraev's system, NL paraphrases are generated at two points in the understanding process: first, after the system has produced an interpretation which reflects lexical and structural aspects of the user's query, and second, at a later stage in the processing, after the specific characteristics of the data to be sought in the database in answer to the user's query have been identified. Both of these paraphrases are produced by the same NL generator.

The TQA system is an NL front-end to databases, which translates NL queries into SQL expressions, which are then evaluated against the database. To allow the user to check the accuracy of the SQL expression, Mueckstein's system QTRANS produces a paraphrase of the SQL expression before it is evaluated, and does so in an interesting way. It parses the SQL expression using SQL's unambiguous context-free grammar.[10] It then maps this SQL parse tree into an English surface structure parse tree using application-independent translation rules in the form of a Knuth Attribute Grammar. During this process, nonterminal nodes of the SQL parse tree are *renamed* (receiving NL grammar labels like Noun_Phrase, Relative_Clause, etc.) *reordered* into pre- and post-modifiers within a Noun_Phrase, *deleted* (suppressing SQL artifacts like join conditions), or *inserted* (providing English function words). In its final step, QTRANS translates the terminal nodes into English and prints out the resulting string.

For example, consider the database query:

What parcels in the R5 zone on Stevens Street have greater than 5000 square feet?

(The database for one TQA application is zoning information for the city of White Plains, NY.)
TQA produces the following SQL expression as its interpretation of the query:

```
SELECT UNIQUE A.JACCN, B.PARAREA
  FROM ZONEF A, PARCFL B
```

[9]Following a technique used in PLANES [70].

[10]Mueckstein has made minor, domain-independent changes to the published SQL grammar, to make it more compatible with the English structures to be generated.

```
WHERE A.JACCN = B.JACCN
  AND B.STN ='Stevens St'
  AND B.PARAREA > 5000
  AND A.ZONE ='R5'
```

This, QTRANS paraphrases as

> Find the account numbers and parcel areas for lots that have the street name Stevens St, a parcel area of greater than 5000 square feet, and zoning code R5.

The important things to notice about this paraphrase are the following:

- It shows the user that the system will be responding with not just a direct answer to the query (i.e., noting just the appropriate parcels) but with additional information as to the area of each.
- It shows the user that the system only knows 'parcels' in terms of their account numbers.
- The phrase 'for lots' is derived from the FROM phrase and represents the 'real world' entity to which the tables ZONEF and PARCFL apply. This must be recorded by hand explicitly for each table and column name in the database.
- The first conjunct in the WHERE clause is suppressed because it is recognized as specifying only the keys compatible for joining.

Mueckstein notes that the difficulties in this approach to paraphrase stem from the need to rearrange the SQL query structure to bring it closer to the user's conceptualization of the domain without being too repetitive or introducing ambiguities. This requires the attribute grammar to be able to compare units across nonadjacent phrases in the SQL query and act accordingly.

Related to this use of paraphrase of a formal language query is the use of NL paraphrase in an automatic programming system developed at ISI [65]. Here the programmer's high-level specifications (which the system will turn into an executable program) are first paraphrased in English so that the programmer can check that s/he has said what s/he intended. (That is, constructs in this high-level language imply particular mapping/dependency relations which the programmar may not be aware of or may have forgotten.) So paraphrase is being used to catch errors before they get in the code—very similar to its use in query systems, to catch bad interpretations before they undergo expensive evaluation against a database.

Before concluding this section, we want to comment on the opposite side of this verification process—allowing the user to verify his/her own understanding of the system's requests or advice. This aspect of user–system interactions has long been overlooked, but is critical for the responsible use of expert systems. Paraphrase, the method employed by systems in resolving ambiguity and verifying their formal interpretation of users' queries, is only one of several methods that are used interactively between people to verify understanding. On the advice giver's side, s/he can explicitly test the recipient's understanding by asking questions or asking the recipient to repeat what s/he (the advice giver) has said. On the recipient's side, s/he may ask the advice giver to verify his/her understanding by *inter alia* paraphrasing the question or advice,

> Expert: OK, under those circumstances, I would not object to see you go back into that for another six months.
> User: So you roll it over, in other words?
> Expert: Right.

by offering back a conclusion s/he draws from the material to be understood

> User: Where is Schedule B?
> Expert: Do you have Schedule A?
> User: Yeah.
> Expert: OK. If you turn it over, there is Schedule B.
> User: Oh. It's on the back of Schedule A?
> Expert: Right.

or by offering back how s/he sees the (general) advice as instantiated to his/own situation

> Expert: You shouldn't mix drugs and alcohol.
> User: You mean I shouldn't take my vitamins with a Bloody Mary?

In user interactions with expert systems, who takes responsibility for initiating this verification will depend on the character of the user population, the goal of the system, and what is being verified. We believe that systems must always provide for users' attempts to verify their understanding of the system's queries. With respect to the system's conclusions and advice, some systems will have to build in tests of users' understanding; others will have to build in additional NL understanding machinery to allow users to initiate verification themselves.

IV. Correcting Misconceptions

Another aspect of the user and system coming to terms with each other for successful interactive problem solving involves their individual world views. Many of the troubles in interaction come from the fact that participants hold different views about the world and either do not realize it or fail to do anything about it. As a result, each may leave the interaction with very different beliefs about what was communicated. In Suchman's terms [63], both user and system must provide, in some way, for the ongoing identification and repair of those troubles.

(In computational research in this area, disparities between the beliefs of user and system have been treated as *misconceptions* on the user's part. That is, the system's view of the world is assumed to be correct. Of course, in actual fact, the user may hold the correct belief, but the important thing is that the disparity be pointed out so that the interaction can be correctly interpreted. In what follows, we shall also refer to these disparities as "misconceptions" to be consistent with the literature.)

Heretofore most of the computational work on recognizing and responding to user misconceptions has been done in the context of NL front-ends to database systems. Notice that in these relatively simple systems, where it is only facts and requests for them that are exchanged, users may be able to recognize a disparity between the system's view of the world and their own: for example, they may be able to tell that the system's answer to their question is wrong or strange. Nevertheless, because the user may not recognize the disparity at all or not recognize it quickly

enough to avoid confusion, people have felt the need to put as much responsibility on the system as they have been able to. The problem is much greater in user interactions with expert systems, where users are likely to have less domain knowledge and expertise than the system and thus are less likely to recognize a disparity. The result is confusion, or worse, misinterpretation. In what follows, we discuss work on recognizing and responding to user misconceptions, first in the context of database question/answering and then in the context of interactions with expert systems.

There are at least two kinds of beliefs that are easy to recognize. One is a belief that a particular description has a referent in the world of interest; i.e., that there is an individual or set which meets the description. Such a belief is revealed when a noun phrase is used in particular contexts. For example, for Q to ask the question "Which French majors failed CIS531 last term?" Q would have to believe that: 1) CIS531 was given last term, 2) there are French majors, and 3) there are French majors who took CIS531 last term. If R does not hold these beliefs as well, it would be uncooperative for R to give a direct answer, since any direct answer, including "None," implicitly confirms (incorrectly) that R holds these beliefs. The other kind of belief concerns what can serve as an argument to a particular relation. Such beliefs are revealed linguistically in terms of the subject and/or object of a verb. For example, the question "Which graduate students have taught CIS531?" shows that Q believes that graduate students can teach CIS531. Again, if R does not hold this belief, it would be uncooperative to give a direct answer, since any answer would implicitly confirm this belief, including the answer "None." In both examples, a direct answer is insufficient, and must be augmented in a response which addresses the disparity.

Computational work on providing such responses has been done by several researchers. Kaplan [33], in his CO-OP system, addresses the misconception that a description of an entity or set has a referent in the system's database, when according to the system's information, it does not. CO-OP looks for a possible misconception if its response to the user's query comes back empty or with a "no" answer. In these cases, CO-OP checks whether all the entities mentioned in the query exist in the database (i.e., correspond to nonempty sets). If one or more does not, CO-OP assumes a misconception on the user's part and responds accordingly; e.g.,

User: Which French majors failed CIS531 last term?
System: I do not know of any French majors.

Mercer [46] addresses both the "extensional" misconceptions (Kaplan's term) that Kaplan considers, and certain misconceptions inferrable from the use of particular lexical items, such as

User: Has John stopped taking CIS531?
System: No, he has not stopped since he hasn't started.

Mercer's system depends on its ability to determine the presuppositions[11] of the answer it plans to give. Where Mercer's and Kaplan's approaches differ is that Kaplan's system essentially infers what the user would have to believe about the world in order to have asked the particular question. If any of those beliefs conflict with the beliefs of the system, corrective information is included in addition to or instead of an answer. Mercer's system, on the other hand, computes the presuppositions of its planned answer because according to Gazdar's interpretation [21] of Grice's Maxim of Quality [24], a speaker should not say anything which has a nontrue presupposition. The difference between the systems would be apparent if CO-OP did not make the "Closed World Assumption." Then CO-OP could not necessarily conclude there were no individuals satisfying a particular description if it could not find them. For example,

User: Do any professors that teach CSE101 teach CSE114?

If no such individuals could be found nor could individuals satisfying the embedded description "professors that teach CSE101," CO-OP would reply

System: I do not know of professors that teach CSE101.

On the other hand, if Mercer's system can prove that there is no professor who teaches CSE114 but cannot prove that its presupposition is false—that there is no professor who teaches CSE101—it will respond

System: No. Moreover I don't even know if any professor teaches CSE101.

Mays [40] and, more recently, Gal [20] address misconceptions that assume that a relationship can hold between two objects or classes, when according to the system's information it cannot. For example

User: Which graduate students have taught CIS531?
System: CIS531 is a graduate course. Only faculty can teach graduate courses.

Mays' system uses a rich data model containing three types of information against which to validate the system's interpretation of the user's query:

- taxonomic information such as a woman is a person, a teacher is a person, etc.;
- mutual exclusion information such as an undergraduate cannot be a graduate student (and vice versa);
- selectional constraints such as the subject of (one sense of) the relation 'teach' must be interpretable as the concept 'teacher' and its object must be interpretable as the concept 'course.'

Every query interpretation is checked against this data model for possible selectional constraint violations. By virtue of the taxonomic and mutual exclusion information, stating the selectional constraints at the highest place in the taxonomy at which they hold captures all and only those concepts that could serve as arguments to a relation. So even though selectional constraints on 'teach' may only be stated in terms of 'teacher' and 'course,' the system can recognize that 'woman' can be the subject of 'teach' (i.e., it is compatible with the concept 'teacher') while 'undergraduate' cannot be (i.e., it is incompatible with the concept 'teacher'). Any violated constraints are taken as evidence of a misconception on the part of the user.

[11] In simple terms, a presupposition of a sentence S is a statement that must be true for either S or its negation to make sense.

Gal's system uses the database system's integrity constraints to achieve a capability which overlaps that of May's system. (Integrity constraints constrain the values that can appear in fields in the database, and are used primarily to maintain consistency when the database is updated.)

By augmenting the system's analysis of the user's query with integrity constraints on the predicates that appear in the query, Gal's system can recognize user misconceptions at two points: 1) before evaluating the query against the database, if an integrity constraint is simply violated by the query, and 2) during query evaluation, if a fact found in the database causes the integrity constraint to be violated. For example, given an integrity constraint like "Professor S teaches courses only in department A," Gal's system would recognize the user's misconception in the query

User: How many courses does professor S teach in department B?

before query evaluation. On the other hand, it would recognize the user's misconception in the query

User: Does Professor S teach 120?

during query evaluation, if it found that course 120 was not in department A.

Gal's system, written in Prolog, uses the same parser to analyze the user's NL queries and to generate corrective NL responses. Additional heuristics are used to decide what to say (i.e., to pass to the parser/generator) in case a corrective response is called for. For example, all violated integrity constraints are always displayed, but if an integrity constraint is only found to be violated during evaluation, the database facts signaling the violation are presented first, followed by the violated constraint, for example:

User: Is John instructed by Professor Smith?
System: No; because Smith teaches only in the English-for-foreigner department; John is American and no American student can be enrolled in the English-for-foreigner department.

Work on recognizing and responding to user misconceptions in the context of expert systems or knowledge based systems is even more preliminary. We note here two such efforts. Work on misconceptions involving confusions about objects and their properties has been done by McCoy [72], [73]. These misconceptions may be revealed through a user's utterance either describing an object in terms of a class it does not belong to, incorrectly attributing some property to it, or ascribing an incorrect value to one of its properties. For example,

System: Do you have any liquid assets?
User: I have a $5000 money market certificate.
System: A money market certificate isn't a liquid asset. Your money is tied up for several years in a money market certificate. Do you mean a money market account?

User: What's the interest rate on this stock?
System: Stocks don't have an interest rate. They may pay a dividend periodically.

One problem with these is whether, in the current context, they actually constitute misconceptions. That is, McCoy shows that the current context/perspective can affect whether two things are judged to be similar or different. Contrasting them in the former situation would reveal a misconception, as would equating them in the latter. For example,

System: I'd like to see you hold that $10000 coming due in a money market fund until mid-March and then get into a longer term money market certificate.
User: Well, I was just wondering, what about a treasury bill instead?
System: That's not a bad idea but it doesn't replace your money market certificate in any way—it's an exact duplicate. They're almost identical types of instruments, so one, as far as I'm concerned, is about the same as another.

User: I'm interested in buying some US Government securities. I was thinking about Money Market Certificates, since they are the same as Treasury Bills.

System: But they're not: they're two very different things. A Treasury Bill is backed by the government. You have to get it from the Federal Reserve. A money market certificate, on the other hand, is backed by the bank that issues it. So one is a government security while the other isn't.

In the first example, both objects are viewed through the perspective of savings instruments, thereby highlighting such attributes as interest rate and maturity date, which treasury bills and money market certificates share. But in the second example, the objects are viewed in terms of their issuer, thereby highlighting such attributes as issuing company and purchase place, which attributes treasury bills and money market certificates do not share. McCoy's approach through perspective is able to identify that contrasting the two in the first example reveals a misconception, as does equating them in the second example.

McCoy presents a variety of correction strategies whose selection in a particular case depends on what the system believes may have led the user to the misconception. For example, in one case, the system will attempt to include in its response

1) a denial of the incorrect classification
2) a statement of the correct classification
3) a concession of similarity by mentioning their common attributes
4) an override to the conceded information by noting features that define the object as belonging to its correct superordinate class.

User: I want to buy some corporate bonds. Should I get preferred?
System: Preferred are not bonds, they are stocks. A preferred stock is like a bond in that both have regular payments of a fixed amount. However, unlike bond payments, the company is under no legal obligation to pay preferred stock payments. In addition, preferred stock gives part ownership in the company.

What is missing from McCoy's system is the machinery

necessary for recognizing when the user has an object-related misconception. It will not be as simple a task as that of recognizing an existential misconception, although we hope that a wide range of cases will be possible. For example, the above case requires the system to interpret the user's second sentence "Should I get preferred?" as a more specific request for information about corporate bonds. The user's goal is to buy corporate bonds. To achieve this goal, s/he needs information or wants advice about corporate bonds. In particular, s/he wants information about a specific type of corporate bonds—preferred. By this chain of interpretation (relating the user's goals and his/her information seeking behavior), the system hypothesizes this misconception on the user's part. This is not a simple type of reasoning, and more work is needed on isolating both easy to recognize object-related misconceptions and the knowledge and reasoning abilities needed to deal with more subtle cases.

On the subject of user plans and goals, this is another area rich in misconceptions that may be more visible in user interactions with expert systems and knowledge-based systems. That is, it is frequently the case that someone has a goal they want to achieve, but that the plan they come up with, which motivates their interaction with someone else, will not achieve that goal. Recognizing this, a human respondent may give information that facilitates the questioner achieving his/her goal, as well as (directly or indirectly) answering the given plan-related question. This kind of behavior is illustrated in the following examples [54]:

Q: Is there a way to tell mail that I'd like to deliver a message after some particular time and/or date?
R: No. I can't really see any easy way to do this either (and guarantee it). If you really wanted to do that, you could submit a batch job to run after a certain time and have it send the message.

Q: What's the combination of the printer room lock?
R: The door's open.

Questioners presume experts know more about a domain than they do, and they expect the above sort of cooperative behavior. If it is not forthcoming, expert systems are likely to mislead their users [31]. If they mislead their users, they are worthless.

This behavior depends on the ability to infer when and how inappropriate plans underlie a user's queries, an ability addressed in [54]. In this work, Pollack develops a model of responding to questions that does not rest on what she has called *the appropriate query assumption*. Abandoning this assumption requires the development of a model of plan inference that is significantly different from the models discussed by Allen [2], Cohen [13], and others. Where their inference procedures are essentially heuristically guided graph searches through the space of valid plans, Pollack's procedure involves R's 1) reasoning about likely differences between R's own beliefs and Q's beliefs and 2) searching through the space of plans that could be produced by such differences. To facilitate this reasoning, Pollack has had to re-analyze the notion of plans, giving a careful account of the mental attitudes entailed by having a plan. Her account derives from the literature in the philosophy of action, especially [23] and [8].

One belief that is claimed to be necessary for a questioner to have a plan is that all the actions in the plan are related either by **generation** [23] or by **enablement**.[12] Consider the first example above. The response can be explained by proposing that the respondent infers that the questioner's plan consists of the following actions:

- telling mail to deliver a message after some particular date and/or time;
- having mail deliver a message after some particular date and/or time;
- having a message delivered after some particular date and/or time.

What the respondent believes is that the questioner believes that the first action *generates* the second, and the second *generates* the third. In the response, the relation that holds between the mentioned action (submitting a batch job) and the inferred goal (having a message delivered after some particular date/time) is also generation, but here, it is the respondent who believes the relation holds. While Pollack does not address the actual information that should be included in a response when the system detects an inappropriate plan underlying a query, it seems clear that an ability to infer such plans is requisite to responding appropriately.

Pollack's analysis can also be applied to handling other types of requests that reflect inappropriate plans. As discussed above, the phenomenon she addresses requires reasoning about the relationship of one action to another, and, more significantly, people's knowledge or beliefs about such relationships: the actions themselves can be treated as opaque. In order to handle the second example above, Pollack's analysis would have to be extended to allow reasoning about people's knowledge or beliefs about objects and their properties and the role that particular objects and their properties play in actions. Reasoning about the relationship between knowledge (primarily of objects) and action has been discussed in [2], [47], [3].

In summary, there are many types of belief disparities that can interfere with the successful transmission of information between user and system. One of the most subtle for expert systems involves the user's belief in the relevance of some concept to his/her question, where for the system, it is irrelevant. Lack of attention to such misconceptions can make it very difficult for the user to understand the system's advice. Examples of this can be found in Cassell's transcripts of doctor's taking medical histories [10]. It is something that the developers of advisory systems will eventually have to deal with, if they are to be used by the lay public.

V. Information Intended to Prevent Misconstrual

In Section IV, we noted that many of the troubles that arise in an interaction come from the fact that participants hold different views about the world and either do not realize it or fail to do anything about it. These different views may lead to utterances being misconstrued. But even when the participants appear to hold the same view of the

[12] Generation corresponds to what is roughly captured as the *by* locution in English: the questioner believes one can have a message delivered *by* telling mail to deliver a message. Enablement corresponds to the relationship captured in the STRIPS model of actions: finding out the combination to the printer room enables one to open the door. Generation and enablement are not the same: telling mail to deliver a message after some particular date and/or time does not alter the current world state in such a way that one can then have mail deliver the message.

world, there are still circumstances where utterances may be misconstrued. To avoid such misunderstandings, responsible human speakers know to augment their utterances with additional information. It is this latter information that we discuss in this section, and what expert systems and knowledge-based systems will require to avoid such misunderstandings as well.

In answering questions, there are at least three bases for a respondent anticipating that the questioner may draw a false conclusion from his/her answer A: *deductive reasoning* involving A and a false belief that the questioner holds; *default reasoning*, where the respondent knows the default does not hold; and *implicature*, where A may suggest, by virtue of its ordinary use, something that is not the case here. The first case is similar to the situation discussed in Section IV, in which the respondent adds information to correct the questioner's misconception. If the respondent does not point out, or better, correct this false belief, the questioner will draw an additional false conclusion from the respondent's answer. The additional information s/he provides in response may point out the false belief or correct it as well [32], [31].

The second case is illustrated by the following hypothetical exchange between user and system:

User: What's the combination of the printer room lock?
System: It's 8-9-8, but Ira changes it every week.

One might call this an "inertial default"—typically, the referent of a definite descriptor stays the same. If the system just answers the user's question ("It's 8-9-8"), s/he may falsely conclude (by default reasoning) that the referent of the definite descriptor "the combination of the printer room lock" will stay 8-9-8. The system provides the additional information ("but Ira changes it every week") to prevent this misconstrual.

The third case, misconstrual based on implicature, is illustrated by the following example:

User: Which ships have a doctor on board?
System: The JFK, the Admiral Nimitz, ...

From the system's answer, the user may conclude that the ships named there have a doctor on board and other ships not named there do not. That is, the user may be led to believe that there is a *non-empty set of ships which do not have a doctor on board*. However, if the named ships are the only ships (and the user does not know it), the system's answer—while correct and useful—is nevertheless misleading: the system should have answered "all of them," which would not have mislead the user to believe in a doctor-less subset of ships.

A fourth possible basis for anticipating a false conclusion from a correct answer is the respondent's belief that people typically believe X (i.e., as opposed to his/her belief that people believe that typically X). Consider a user interacting with an employment database, checking out a job that looks interesting. S/he asks

User: What's the salary?
System: There isn't any.

Because of this, the user rejects the job and moves on to consider others. The system's answer may be factually correct (i.e., there is no salary entry for the job), but the user's conclusion may be false: that is, the job may not pay a salary, but remuneration may come in other ways; e.g., tips, commission, stock options, etc.

To anticipate and avoid such a false conclusion, the system could reason about what people typically believe, and if that does not match the current situation, it could inform the user. That is, suppose it believes that people typically will not take nonpaying jobs. It also believes that people typically believe that if a job does not pay a salary, it is a nonpaying job. (Here, SB stands for "System believes" and UB, for "User believes".)

(SB (*typically* U not take a nonpaying job))
(SB (*typically* UB (no salary ⇒ nonpaying job)).

The system, on the other hand, believes that all jobs either pay a salary or have tips or offer commission on sales or provide stock options or other free benefits or are nonpaying. Moreover, the system knows that this job offers a 20-percent commission on sales.

(SB∀x : job. x pays salary or x has tips or x offers commission ... or x is nonpaying job)
(SB JOB617 not pay salary)
(SB JOB617 offers commission)

Given the system's own beliefs and beliefs about user's typical beliefs, the system should be able to anticipate that the user will not take this job (JOB617) because it is nonpaying. However, it knows that while the job does not pay a salary, it does pay commission. Thus the system should also describe what reimbursement it provides, to prevent any misconstrual.

System: There is no salary, but there is a 20-percent commission on every squash racquet you sell.

The principles that guide this behavior are Grice's Maxim of Quantity [24]:

> Make your contribution as informative as is required for the current purposes of the exchange. Do not make your contribution more informative than is required for the current purposes of the exchange.

and Joshi's modification to Grice's Maxim of Quality [30]:

> If you, the speaker, plan to say anything which may imply for the hearer something that you believe to be false, then provide further information to block it.

What this modified Quality Maxim provides is a criterion for the level of informativeness needed to satisfy the Quantity Maxim: enough must be said so that the hearer does not draw a false conclusion.

Hirschberg has investigated a class of indirect and modified direct responses to yes–no questions in her work on scalar implicatures [28]. It is possible to view a large class of the responses she considers as attempts by cooperative speakers to block potential false inferences which hearers might otherwise (wrongly) infer to be implicatures arising from a direct response—while also providing information from which direct response can be derived [28]. We can illustrate this by the following example.

A: Has Mary had her medication?
B: She has taken the Excedrin.

A simple direct response "No" by B, although correct, would be misleading because it licenses A to draw the false inference that Mary has not taken any of her medication.

B's response above blocks this inference. Note also B indirectly conveys to A the fact that Mary has not taken all of her medication [28].

The major assumption of this work in general is that one can limit the amount of reasoning that R is responsible for doing, to detect a possible reason for misconstrual. This is an important assumption, and further work is necessary to show that such constraining can be done in some principled manner.

VI. Adapting the Interaction to Individual Users

As noted earlier, system-generated justifications and explanations can eliminate potential troubles in user–system interactions before they start. So too can adapting the interactions to the abilities and interactional style of the user. Although this adaptive behavior is not unique to systems employing an NL interface, it is strongly evident in normal human interaction and will be expected by users of systems with sophisticated NL interfaces. The issues we will discuss are 1) deciding what questions to ask and not to ask of the user; 2) interpreting a "don't know" answer; and 3) enlarging the range of acceptable answers.

A. Deciding What to Ask

Existing interactive expert systems employ a number of techniques to guide their search for appropriate conclusions. For the most part, they employ strategies which take into account such preferences as:

- prefer rules which can yield definite conclusions over indefinite ones (e.g., MYCIN's *unitypath* heuristic) [59];
- prefer rules whose conclusions can have the greatest impact on the ultimate goals of the system (e.g., the rule selection function in PROSPECTOR [17]);
- prefer rules which do not require asking the user any questions over those which do (e.g., as in MYCIN and ARBY [41]).

Most expert systems fail, however, to take into account any factors which depend on the individual they are interacting with—for example, the user's ability to understand a question or to give a reliable answer. This section discusses some of the ways that an expert system can improve its interaction with a user by incorporating an explicit model of the user's knowledge and beliefs.

Interactive expert systems vary as to when they ask for information and when they rely on their own deductions. However, in this decision, they ignore the user's ability to understand and respond reliably. For example, in PROSPECTOR [17], goals are simply marked as being either "askable" or "unaskable" (never both). In MYCIN [59], the user is only asked for information if either the system's attempt to deduce a subgoal fails; i.e., if no rules were applicable or if the applicable rules were too weak or offset each other; or the user's answer would be conclusive (e.g., lab results). In KNOBS [18], a system for assisting users in mission planning, the user is only asked for preferences, not facts. If the user prefers not to answer at any point, s/he can turn over control to the system and let it compute an appropriate value.

Any attempt to customize a system's interaction to the current user must allow for wrong guesses. The user may not be able to answer its questions or will answer them incorrectly or will find them annoying. Thus customization has several aspects—1) *deciding* what question to ask next; 2) *recovering* from a wrong decision (i.e., from having asked a "bad") question; and 3) *modifying* subsequent decisions about what sort of questions to ask.

One approach might be to evaluate strategies according to how much work is required of the user to provide the information requested of him/her. This can be factored into:

1) How much work is required to understand the question?
2) How much work is required to acquire the information?
3) How much work is required to communicate the information to the system?

A strategy that required no further information from the user would receive an "easy" rating. Somewhat harder would be a strategy that required the user to make an observation, and harder still would be a strategy that required a test. Much harder would be one that needed to access the user's prior knowledge for the information. For example, if a piece of equipment failed to work, an obvious question to ask is "Is it plugged in?" This is simple to ascertain by observation and has a high payoff. Somewhat harder to answer because it involves a test, but again something with a high payoff, is the question "Is the battery working?"[13]

Of course there might be several alternative procedures the user could employ in acquiring the information the system wants, each of different difficulty for him/her, each requiring somewhat different resources. While the system's evaluation of a strategy might be based on the assumption that the user can and will use the easiest of these procedures, more refined evaluations might take into account the *resources available* to the particular user as well. (This information about alternative procedures—their level of difficulty and resource requirements would also be useful for certain cases where a user cannot answer the system's question, as will be discussed in the next subsection.)

A strategy evaluation based solely on how much work is required of the user would not be sufficient however. Another factor in the system's choice of reasoning strategy must be its *a priori* beliefs about the reliability of the user's information. The system should prefer a line of reasoning which depends on facts it believes the user can supply reliably over one which it believes the user can supply with less reliability.

For example, consider a system taking a patient's medical history. If it needs to know whether the patient has a drinking problem it can either ask the question directly (e.g. "Are you an alcoholic?") or ask a set of questions from

[13] This resembles somewhat the strategies embodied in INTERNIST [55]: when simply trying to eliminate a hypothesis from a large number of possible ones, INTERNIST limits its questions to information obtainable via history or physical exam. Later, when trying to discriminate between a small number of contending hypotheses, INTERNIST may request information which comes from more "costly" procedures. Finally, when trying to confirm a particular hypothesis, there are no constraints on the data it may request: biopsies may be required, etc.

which it can conclude the answer (e.g. "Do you drink socially?" "Do you drink at home?" "Would you say you drink as much as a bottle a day?"[14], etc). Often such an indirect approach can provide more reliable data.

B. When the User Cannot Answer

When the system does choose to go to the user for information, the user may still only be able to answer "I don't know." A flexible system should be able to take this response, hypothesize the underlying reasons for it, and act to help the user out of his/her predicament. This section describes some of the underlying reasons why a person cannot (or will not) answer a question, each of which calls for a slightly different response on the system's part. Note that no matter what the reason, the user may still respond with a simple "*I don't know.*"

The user *may not understand* the question; e.g., it may contain unfamiliar terms or concepts. Here the system should have one of two possible responses: either it should be able to supply the user with definitions of the terms or concepts s/he is unfamiliar with (cf. Section II) or it should be able to rephrase the question so as not to use them.

The user *may not be sure* that the question means the same to the system as it does to him/her. In this case, the system should be able to generate a paraphrase of the question that clarifies it or allow the user to verify his/her understanding (cf. Section III).

The user may understand the question but *not know how* to go about determining the answer. In this case, the system should be able to suggest one or more procedures for doing so.

The user may understand the question but *may not remember* the information needed for an answer. In this case, the system should be able to ask the user for information that provides strongly suggestive clues to the information it needs.

The user may understand the question but *may not have at hand* the information needed for determining an answer (e.g., lab results). The system should be able to figure out whether it might be able to perform some preliminary reasoning without the missing information and finish things off when it is provided.

The user *may not know why* the system wants the information and will not divulge it until s/he does. In this case, the question needs to be expanded to include the system's reason for asking it.

The user *may not believe* the requested information is *relevant* to solving the problem and is trying to force the expert to adopt another line of reasoning. In this case, the system should be able to identify and pursue an alternative strategy, if one exists. If not, it should be able to explain to the user why it cannot continue. The same holds if it is the case that the user does not wish to divulge the information.

Since its strategy for possible success differs in each case, a system must be able to 1) determine why the user did not answer and 2) offer an appropriate response to help the user formulate an answer. There are several techniques it can use in achieving the first goal:

- It can provide the user with a way to express the problem directly (e.g., through a menu or a set of predefined key words like PARAPHRASE).
- It can use a model of the individual user to determine the most likely problem (e.g., some terms are very likely to be unknown by certain classes of users).
- It can interpret a user's NL response against the set of possibilities

Once the problem has been found, there are also several possible response techniques. If the user does not understand terms in the question, then the techniques discussed in Section II are applicable. If the user is uncertain of the overall meaning of the question, then the paraphrasing techniques of Section III are in order. Some other useful techniques to implement include:

1) Procedures for the user's acquiring the information, annotated by the level of resources, time and sophistication needed to carry them out, as in these exchanges:

System: Does eating certain foods cause a problem for you?
User: HOW_KNOW?
System: Do you get a rash when you eat them, or become short of breath or vomit?

System: Is it an ASCII file?
User: HOW_KNOW?
System: Is it readable when printed on your terminal?

2) "Script"-like information relating events of interest with events that are perhaps more easily remembered by the user [35], as is suggested by this interaction:

System: Have you had a tetanus shot in the last two years?
User: DON'T_RECALL
System: Were you bitten by a dog, cat, or some other animal in the last two years?
User: Yes.
System: Were you treated for the bite?
User: Yes.
System: Were you given a shot as part of the treatment?
User: Yes.
System: Then that was probably a tetanus shot.

3) Teleological information concerning the reasons for making the request, as is shown in this:

System: How tall are you?
User: WHY?
System: We would like to know whether your weight is right for your height.

4) Ability to do case analysis reasoning; i.e., considering the effect on the outcome, under the alternative assumption of each possible value or class of values. (It is possible that it will not make a difference which value is assumed.); e.g.:

System: What is your mother's Rh factor?

[14]This last question is interesting in its own right, as Cassell [10] notes that one is more likely to get a reliable answer to a question that lets the patient off the hook. "No, not that much, only 5 or 6 shots."

User: DON'T_KNOW
System: Could you ask her? We'll continue now without it.

System: Has anyone in your family committed suicide?
User: SKIP_IT

Note that most of these examples show a system providing another question rather than a definition for the unknown or unshared term or a procedure for determining the answer. The reason for this is that it constrains the subsequent interaction, so that the user knows what is expected of him/her, thereby avoiding some problems of interpretation.

C. Increased Leeway in User's Responses

We have been talking all along about enlarging the *range* of interactions that responsible users need to be able to have with expert and knowledge based systems to ensure that the advice the user receives is appropriate and is understood correctly. As the range widens, the entire interaction becomes more natural. As a result, users may become frustrated if their normal interactive behavior is curtailed. One place that it might be is in responding to the system's requests for information.

People vary in the amount and kind of information they are used to providing in response to questions. They are frustrated if they must act differently. For example, van Katwijk *et al.* [68] report experiments comparing three sources of information on local train schedules—a person, a taped recording, and a simulated computerized information system. What upset people most about the latter was its refusal to process any information that was not explicitly requested. Obviously, the person volunteered it because s/he felt the system would need it. Frame-based systems such as KNOBS [18] and GUS [6], an interactive travel assistant developed at XEROX, permit such behavior, and other systems must do so as well. Here we indicate several other ways (besides *volunteering additional information*) in which an expert system should allow a user more flexibility in answering a question.

Offering Facts from which an Answer can be Deduced: Often the user provides an indirect answer, in the belief that the system can and will deduce a direct answer from it. We have identified four situation in which this occurs.

1) The user is *unable* to determine an answer to the question but has what s/he believes to be information from which the system can deduce an answer, as is shown in the following examples.

System: What is your employee classification: A-1, A-2, or A-3?
User: I'm an assistant professor in Oriental Studies.
System: All faculty members are A-1 employees, thank you.

System: Are you a senior citizen?
User: I'm 62 years old.

Of course the user can be wrong, and the information s/he offers either inadequate or irrelevant, as in this exchange.

Systems: What is your employee classification: A-1, A-2, or A-3?
User: I've been here for over 5 years.
System: Sorry, could you tell me either your job title or position?

2) The user is *able but unwilling* to perform the computations necessary for answering the question, as in:

System: What is your yearly salary?
User: $1840 per month.

3) The user goes beyond a direct literal answer to provide a more precise and hence possibly more informative answer, as in the second of the following two responses:

System: Are you 65 or older?
User 1: Yes.
User 2: I'm 72.

4) The user answers a slightly different question because a direct answer to the given question would be *logically correct but misleading*. This is in the same behavior we discussed in Section V in terms of the system modifying its correct but potentially misleading direct answers. Users must be free to do the same, for example:

System: Did you delete all the KLONE files?
User 1: Yes.
User 2: I deleted the .LISP files.

System: Did you delete all the KLONE files?
User 1: No.
User 2: I deleted the .LISP files.

If the user were confined to answering just yes or no, the system would be misled: if the user just answered "no," the system might think that no files were deleted (which is incorrect, in the second case). If it just answered "yes," the system might think that just those files had been deleted (which is incorrect, in the first case).

Hedging: Often a user may wish to hedge on his answer, say by attaching a certainty measure, offering a range of values, or providing a disjunctive answer. Many current expert systems provide for the first case. On the other hand, the possibility of hedging by specifying a disjunctive answer or a range of values has not been explored to any extent. There are several problems with allowing a user these possibilities. One involves discriminating between an *imprecise single value* and a *precise range of values*. Contrast the following two interactions:

System: How tall is your wife?
User: Between 5'2" and 5'4".

System: How much does a Volvo cost?
User: Between 12K and 18K.

In the first the user is clearly giving a range in which the single answer lies. In the second, s/he is more likely to be indicating that **a Volvo** specifies a set of objects, the costs of which fall in the range given. Determining which of these two cases the user intends requires a model of how the user views the object and attribute.

Deferring an Answer: The user may wish to defer answering a question. This differs from declining to answer (i.e., the SKIP_IT response discussed earlier) in signaling that the user wants to supply the information later, but not now. In an expert system that responds to user preferences (e.g., any kind of planning assistant), one reason for defer-

ring is that for the user the answer is *conditional* on something that has not been resolved yet. A deferral gives the user a method to control the reasoning process of the expert system by forcing it to pursue lines of reasoning which do not require the requested information. For example, consider an expert system which recommends restaurants. The system may ask about cost before cuisine, whereas for the user what s/he's prepared to spend depends on the type of cuisine; e.g.:

System:	How much do you want to spend?
User:	DEFER
System	Do you prefer a restaurant within walking distance?
User:	Yes.
System:	What kinds of cuisine do you prefer?

In order to handle the kinds of responses to direct questions that we have shown here, a system needs to determine 1) the *relationship* that holds between question and response; 2) the *answer* to the given question; and 3) the *additional information*, if any, that the response provides.

VII. Conclusion

Artificial Experts, often called Knowledge Based Systems, Expert Systems, or Advisory Systems, involve more than simple question answering. NL interfaces for such systems do more than identify or retrieve facts. They must support a variety of cooperative behaviors for the interactions to be successful. The user and the system have to come to terms with each other, they must understand each other's terminology, problems, analyses, and recommendations. The user cannot be expected to come to the system with an already well-formed problem and show an accurate understanding of the system's capabilities and terminology, or even a good understanding of the domain. In this paper we have discussed specific NL capabilities that support aspects of such cooperative behaviors. We have focussed exclusively on such interactions (and not on the fact retrieval aspect of question-answering) because we believe that NL interfaces, of very rich functionality, are critical to the effective use of artificial experts. These aspects of interaction are precisely those where NL interfaces really pay off and will do so even more in the future.

Acknowledgment

The authors like to thank their referee for his or her excellent comments. They would also like to thank those people whose work they have reported on here: Kathy McKeown, R. Rubinoff, Eva Mueckstein, J. Kaplan, B. Mercer, E. Mays, Annie Gal, Kathy McCoy, Martha Pollack, and Julia Hirschberg. Finally, they would like to thank MaryAngela Papalaskaris for suggesting this title.

References

[1] J. Allen and C. R. Perrault, "Analyzing intention in utterances," *Artificial Intell.*, vol. 15, pp. 143–178, 1980.

[2] J. Allen, "Recognizing intentions from natural language utterances," in M. Brady, Ed., *Computational Models of Discourse*. Cambridge, MA: MIT Press, 1982.

[3] D. Appelt, *Planning English Sentences*. Cambridge, England: Cambridge University Press, 1985.

[4] M. Bates and R. J. Bobrow, "Natural language interfaces: What's here, what's coming, and who needs it," in W. Reitman, Ed. *Artificial Applications for Business*. Norwood, NJ: Ablex Publishing, 1984.

[5] ———, "A transportable natural language interface," in *Proc. 6th Ann. Int. SIGIR Conf. on Research and Development in Information Retrieval*, ACM, 1983.

[6] D. Bobrow, R. Kaplan, D. Norman, H. Thompson, and T. Winograd, "GUS, A frame driven dialog system," *Artificial Intell.*, vol. 8, 1977.

[7] B. K. Boguraev and K. Sparck-Jones, "A natural language front end to databases with evaluative feedback," in G. Gardarin and E. Gelenbe, Eds., *New Applications of Data Bases*. London, England: Academic Press, 1984, pp. 159–182.

[8] M. Bratman, "Taking plans seriously," *Social Theory and Practice*, vol. 9, pp. 271–287, 1983.

[9] S. Carberry, "A pragmatics-based approach to understanding intersentential ellipses," in *Proc. 23rd Ann. Meet.* (Association for Computational Linguistics, University of Chicago, Chicago IL, July, 1985), pp. 188–197.

[10] E. J. Cassell, *Talking with Patients. Volume I: The Theory of Doctor Patient Communication*. Cambridge, MA: MIT Press, 1985.

[11] W. Clancey and R. Letsinger, "NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching," in W. Clancey and E. Shortliffe, Eds., *Readings in Medical Artificial Intelligence: The First Decade*. Reading, MA: Addison-Wesley, 1984, pp. 361–381.

[12] E. F. Codd, "Seven steps to rendezvous with the casual user," in J. W. Klimbie and K. L. Koffeman, Eds., *Data Base Management*. Amsterdam, The Netherlands: North-Holland, 1974, pp. 179–199.

[13] P. Cohen, "On knowing what to say: Planning speech acts," Dep. of Computer Science, Univ. of Toronto, Toronto, Ont. Canada, Tech. Rep. 118, Jan. 1978.

[14] R. Creagh-Osborne, *This is Sailing*. Boston, MA: SAIL Publ., 1972.

[15] F. J. Damerau, "Operating statistics for the transformational question answering system, *AJCL*, vol. 7, no. 1, pp. 30–42, 1981.

[16] R. Davis, "Interactive transfer of expertise," *Artificial Intell.*, vol. 12, no. 2, pp. 121–157, 1979.

[17] R. Duda, J. Gaschnig, and P. Hart, "Model design in the PROSPECTOR consultant system for mineral exploration," in D. Michie, Ed., *Expert Systems in the Micro-electronic Age*. Edinburgh, UK: Edinburgh Univ. Press, 1979.

[18] C. Engleman, E. Scarl, and C. Berg, "Interactive frame instantiation," in *Proc. 1st Nat. Conf. on Artificial Intelligence (AAAI)*(Stanford CA, 1980).

[19] S. Feiner, "APEX: An experiment in the automated creation of pictorial explanations," *IEEE Comput. Graphics and Applications*, vol. 5, pp. 29–37, Nov. 1985.

[20] A. Gal and J. Minker, "A natural language database interface that provides cooperative answers," in *Proc. 2nd IEEE Ann. Conf. on Artificial Intelligence Applications* (Miami, FL, Dec., 1985), pp. 352–357.

[21] G. Gazdar, "A solution to the projection problem," in C.-K Oh and D. Dinneen, Eds., *Syntax and Semantics*. New York: Academic Press, 1979, pp. 57–90.

[22] J. Ginsparg, "A robust portable natural language data base interface," in *Proc. Conf. on Applied Natural Language Processing*, pp. 25–30, ACL, 1983.

[23] A. Goldman, *A Theory of Human Action*. Englewood Cliffs, NJ: Prentice-Hall, Englewood Cliffs, NJ: Prentice-Hall, 1970.

[24] H. P. Grice, "Logic and conversation," in P. Cole and J. L. Morgan, Eds., *Syntax and Semantics*. New York: Academic Press, 1975.

[25] L. Harris, "The advantages of natural language programming," in Sime and Coombs, Eds., *Designing for Human-Computer Communication*. London, England: Academic Press, 1983, pp. 73–85.

[26] D. Hasling, W. Clancey, and G. Rennels, "Strategic explana-

tions for a diagnostic consultation system," *Int. J. Man-Machine Studies* vol. 20, pp. 3–20, Jan. 1984.
[27] G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a natural language interface to complex data," *ACM Trans. Database Syst.*, vol. 3, no. 2, pp. 105–147, June 1978.
[28] J. Hirschberg, "The theory of scalar implicature," Tech. Rep. MS-CIS-85-56, Dep. Computer and Information Science, Univ. of Pennsylvania, Philadelphia, Dec. 1985.
[29] J. Hollan, E. Hutchins, and L. Weitzman, "STEAMER: An interactive inspectable simulation-based training system," *AI Mag.* vol. 5, no. 2, pp. 15–28, Summer 1984.
[30] A. K. Joshi, "Mutual beliefs in question answering systems," in N. Smith, Ed., *Mutual Belief*. New York: Academic Press, 1982.
[31] A. K. Joshi, B. Webber, and R. Weischedel, "Living up to expectations: Computing expert responses," in Proc. *AAAI-84* (Austin TX, Aug. 1984).
[32] ———, "Preventing false inferences," in *Proc. COLING-84 (Stanford, CA, July*, 1984).
[33] S. J. Kaplan, "Cooperative responses from a portable natural language database query system," in M. Brady, Ed., *Computational Models of Discourse*. Cambridge, MA: MIT Press, 1982.
[34] M. Kay, "Functional grammar," in *Proc. 5th Ann. Meet. of the BLS* (Berkeley Linguistics Society, Berkeley, CA, 1979).
[35] J. K. Kolodner, "Organization and retrieval in a conceptual memory for events or CONS54, where are you?" in *Proc. IJCAI-81*, pp. 227–233. August. 1981.
[36] D. Lanam, R. Letsinger, S. Rosenberg, P. Huyun, and M. Lemon, "Guide to the heuristic programming and representation language. Part 1: Frames," Tech. Rep. AT-MEMO-83-3, Application and Technology Lab., Computer Res. Cen., Hewlett-Packard, Jan. 1984.
[37] W. G. Lehnert and S. P. Shwartz, "EXPLORER: A natural language processing system for oil exploration," in Proc. Conf. on Applied Natural Language Processing, *ACL*, 1983.
[38] D. Litman and J. Allen, "A plan recognition model for clarification subdialogues," in *Proc. 10th Int. Conf. on Computational Linguistics, COLING-84* (Stanford, CA, July, 1984), pp. 302–311.
[39] P. Martin, D. Appelt, and F. Pereira, "Transportability and generality in a natural language interface system," in *Proc. 8th Int. Conf. on Artificial Intelligence* (Karlsruhe, West Germany, Aug. 1983), pp. 573–578.
[40] E. Mays, "Failures in natural language system: Application to data base query systems," in *Proc. 1st Nat. Conf. on Artificial Intelligence (AAAI)* (Stanford, CA, Aug. 1980).
[41] D. McDermott and R. Brooks, "ARBY: Diagnosis with shallow causal model," in *Proc. Nat. Conf. on Artificial Intelligence* (Carnegie-Mellon Univ., Pittsburgh, PA, Aug. 1982), pp. 314–318.
[42] D. McDonald, "Dependency directed control: Its implications for natural language generation," in N. Cercone, Ed., *Computational Linguistics*. New York: Pergamon, 1983, pp. 111–130.
[43] K. McKeown, "Paraphrasing using given and new information in a question-answer system," in *Proc. 17th Ann. Meet. of the ACL* (Association for Computational Linguistics, Aug. 1979), pp. 67–72. Also in *Amer. J. Comp. Ling.*, vol. 9, no. 1, pp. 1–11, Jan.–Mar. 1983.
[44] K. McKeown, M. Wish, and K. Matthews, "Tailoring explanations for the user," in *Proc. 1985 Conf.* (Int. Joint Conf. on Artificial Intelligence, Los Angeles, CA, Aug. 1985).
[45] K. McKeown, *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge, England, Cambridge Univ. Press, 1985.
[46] R. Mercer and R. Rosenberg, "Generating corrective answers by computing presuppositions of answers, not of questions," in *Proc. 1984 Conf.* (Canadian Society for Computational Studies of Intelligence, University of Western Ontario, London, Ont., May, 1984).
[47] R. C. Moore, "A formal theory of knowledge and action," in R. C. Moore and J. Hobbs, Eds., *Formal Theories of the Commonsense World*. Norwood, NJ: Ablex Publishing, 1984.
[48] M. Moser, "Domain dependent semantic acquisition," in *Proc. 1st Conf. on Artificial Intelligence Applications*, R. M. Haralick, Chairman (Denver, CO, Dec. 1984); published by the IEEE Computer Soc. Press.
[49] E.-M. Mueckstein, "Q-Trans: Query translation into English, in *Proc. 8th Int. Joint Conf. on Artificial Intelligence* (IJCAI, Karlsruhe, West Germany, Aug. 1983), pp. 660–662.
[50] R. Neches, W. Swartout, and J. Moore, "Explainable (and maintainable) expert systems," in *Proc. 1985 Conf.* (Int. Joint Conf. on Artificial Intelligence, Los Angeles, CA, Aug. 1985).
[51] W. Ogden and S. Brooks, "Query languages for the casual user," in *CHI '83 Conf. Proc.* (ACM Special Interest Group on Computer and Human Interaction, Boston, MA, Dec. 1983).
[52] C. R. Perrault and B. J. Grosz, "Natural language interfaces," *Annu. Rev. in Comput. Sci.* (Annual Reviews Inc., Palo Alto, CA), vol. 1, pp. 47–82, 1986.
[53] S. J. Petrick, "Transformational analysis," in R. Rustin, Ed., *Natural Language Processing*. New York: Algorithmics Press, 1973, pp. 27–41.
[54] M. Pollack, "Information sought and information provided," in *Proc. CHI '85* (Assoc. for Computing Machinery (ACM), San Francisco, CA, Apr. 1985), pp. 155–160.
[55] H. E. Pople, Jr., "Heuristic methods for imposing structure on ill-structured problems," in P. Szolovits, Ed., *Artificial Intelligence in Medicine*. Boulder, CO: Westview Press, 1982, ch. 5, pp. 119–190.
[56] R. Rubinoff, "Explaining concepts in expert systems: The CLEAR system," in *Proc. 2nd Conf. on Artificial Intelligence Applications* (IEEE, Miami, FL., Dec. 1985).
[57] ———, "Adapting MUMBLE," in *Proc. 24th Ann. Meet.*, Tech. Rep. MS-CIS-86-22, Dep. Comput. and Inform. Sci., Univ. of Pennsylvania, Philadelphia, PA, Apr. 1986.
[58] M. Sergot, "A query-the-user facility for logic programming," in P. Degano and E. Sandewall, Eds., *Integrated Interactive Computing Systems*. Amsterdam, The Netherlands: North-Holland, 1983, pp. 27–41.
[59] E. Shortliffe, *Computer-based Medical Consultations: MYCIN*. New York: Elsevier, 1976.
[60] ———, "Details of the consultation system," in B. Buchanan and E. H. Shortliffe, Eds., *Rule-Based Expert Systems*. Reading, MA: Addison-Wesley, 1984, pp. 78–132.
[61] C. Sidner, "What the speaker means: The recognition of speakers' plans in discourse," *Int. J. Comput. Math.*, vol. 9, pp. 71–82, 1983.
[62] P. Slattery, "PATHEX: A diagnostic expert system for medical pathology," Master's thesis, University of Pennsylvania, Philadelphia, Aug. 1985.
[63] L. Suchman, "Common sense in interface design," in *Conf. on Work and New Technology*, NAWW, Oct. 1982.
[64] W. Swartout, "XPLAIN: A system for creating and explaining expert consulting systems," *Artificial Intell.*, vol. 21, no. 3, pp. 285–325, Sept. 1983.
[65] ———, "The GIST behavior explainer," in *Proc. Nat. Conf. on Artificial Intelligence* (AAAI, Aug. 1983), pp. 402–407.
[66] H. R. Tennant, K. M. Ross, R. M. Saenz, C. W. Thompson, and J. R. Miller, "Menu-based natural language understanding," in *Proc. 21st Ann. Meet.* (ACL, Cambridge, MA, 1983), pp. 151–158.
[67] F. B. Thompson and B. H. Thompson, "Practical natural language processing: The REL system Prototype," in M. Rubinoff and M. C. Yovits, Eds., *Advances in Computers*, New York: Academic Press 1975, pp. 109–168.
[68] A. van Katwijk, F. van Nes, H. Bunt, H. Muller, F. Leopold, "Naive subjects interacting with a conversing information system," Tech. Rep. IPO Ann. Progress Rep., Eindhoven, The Netherlands, 1979.
[69] J. Wallis and E. Shortliffe, "Customized explanations using causal knowledge," in B. Buchanan and E. H. Shortliffe, Eds., *Rule-Based Expert Systems*. Reading, MA: Addison-Wesley, 1984.
[70] D. L. Waltz, "An English language question answering system for a large relational database," *CACM* vol. 21, no. 7, pp. 526–539, July 1978.
[71] W. Woods, "Semantics and quantification in natural language question answering," in M. Yovits, Ed., *Advances in Computers, Vol. 17*. New York: Academic Press, 1978, pp. 2–87.
[72] K. McCoy, "Correcting misconceptions: What to say," in *Proc. Conf. on Human Factors in Computing Systems—CHI '83* (Cambridge, MA, Dec. 1983).
[73] ———, "Correcting object-related misconceptions, "Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, Tech. Rep. MS-CIS-85-57, 1985.